Computers Math. Applic. Vol. 23, No. 1, pp. 103–110, 1992
Printed in Great Britain.

0097-4943/92 $5.00 + 0.00

THE COMPUTATION OF C^k SPLINE FUNCTIONS

MARC ROUFF
Laboratoire de Génie Électrique de Paris, C.N.R.S. - E.S.E.
Plateau du Moulon F 91192 - Gif sur Yvette, France

(Received July 1991)

Abstract—We present in this paper an algorithm for computing C^k spline functions, i.e., spline functions which generate k time continuous and derivable approximations. These functions seem to be very efficient in several applied nonlinear differential problems, like nonlinear control, or nonlinear simulation.

1. INTRODUCTION

The idea of computing an approximated solution of a differential problem is not new. Since Fourier and his heat equation, various functions have been proposed for this task, trigonometric functions of course [1], exponential functions, various polynomials functions like Legendre, Chebitcheff, Laguerre [2–6] or spline functions [7–10], and some discontinous functions like Walsh functions or multi-scaled functions [11–18]. Roughly, we can classify these functions into two classes: the orthogonal functions and the non-orthogonal functions.

The use of orthogonal functions as trigonometric functions, or orthogonal polynomials, are very nice for linear differential problems because many theories can then be written very simply in terms of orthogonality or scalar product in an Hilbert space. The non-orthogonal functions are generally used and developed for nonlinear differential problems. Its associated theories are based on means square consideration or on the minimisation of a canonical functionnal distance in a specified set of functions [19–22].

We present in this article an algorithm for computing C^k spline functions (i.e., spline functions which generate k time continuous and derivable approximations). These functions seem to be a very good tool for various applied nonlinear differential problems, and particularly in nonlinear control. We can give as examples of potential applications, computations in nonlinear optimal control, nonlinear two points boundary values problems, functional differential equations problems, delayed differential equations, shock problems, etc. The extension to several variables permits the investigation of partial differential problems.

After reviewing the C^k spline concept we will give the algorithm. Spline functions for $k = 2$ are given, and the program written in Reduce [23] is in the Appendix.

2. SPLINE FUNCTIONS

The C^k spline principle is to approximate a C^∞ or analytical multi-variate function $u(x_1, x_2, x_3, \ldots, x_n)$ define on an open Ω, by a set of local multi-variate polynomials $\{S^D\}$ of order D, each polynomial or each subset of polynomials (see (2) and (4)) $\{S_i^D(x_1, x_2, \ldots, x_n)\}$ of $\{S^D\}$ is defined on a subopen ω_i of Ω. We must have $\bigcup_i \omega_i = \Omega$ and $\bigcap_i \omega_i \neq \emptyset$.

The degree D of each $\{S_i^D\}$ depends obviously on the choice of the criterion associated to the approximation. In this paper we have chosen a k derivable continuity criterion.

2.1. k Derivable Continuity Criterion and One-Dimensional Spline Functions

Given an open $\Omega = [0, X]$ with I+1 equally spaced nodes called, x_{00}, x_{01}, x_{02}, ..., x_{0I}, and let us consider the set $\{P^D(x)\}$, of the local approximation of a fixed C^∞ function u(x) defined on Ω. Each polynomial, $P_i^D(x)$ is defined on $\omega_i^P = [x_{0(i-1)}, x_{0i}]$ (here we have $\bigcap_i \omega_i^P = \emptyset$) with $i \in [1, I]$.

The k derivative continuity criterion is expressed by

$$\left.\frac{d^\nu P_i^D(x)}{dx^\nu}\right|_{x=x_{0(i-1)}} = u^{(\nu)}[i-1], \text{ and}$$
$$\left.\frac{d^\nu P_i^D(x)}{dx^\nu}\right|_{x=x_{0i}} = u^{(\nu)}[i] \tag{1}$$

with

$$u^{(\nu)}[i-1] = \left.\frac{d^\nu u(x)}{dx^\nu}\right|_{x=x_{0(i-1)}}, \qquad u^{(\nu)}[i] = \left.\frac{d^\nu u(x)}{d\,x^\nu}\right|_{x=x_{0i}},$$

$$i \in [1, I], \text{ and } \nu \in [0, k].$$

These $2(k+1)$ equations define $2(k+1)$ unknowns, then a k derivable continuity criterion leads to a set $\{P^D\}$ with $D = 2k+1$.

For an easier algebraic manipulation and computation, we can define $\tilde{u}_k(x)$ the C^k approximation of $u(x)$ by,

$$\tilde{u}_k(x) = \sum_{i=0}^{I} \left[u^{(0)}[i]\, S_i^{k,0}(x) + u^{(1)}[i]\, S_i^{k,1}(x) + \cdots + u^{(k)}[i]\, S_i^{k,k}\right] \tag{2}$$

where $\left\{S_i^{k,0}(x), S_i^{k,1}(x), \ldots, S_i^{k,k}(x)\right\}$, is the subset $\{S_i^D\}$, of $k+1$ polynomials, of $\{S^D\}$ with $D = 2k+1$, defined on $\omega_i^s = [x_{0(i-1)}, x_{0(i+1)}]$ for $i \in [1, I-1]$, and on $\omega_0^s = [x_{00}, x_{01}]$ and $\omega_I^s = [x_{0(I-1)}, x_{0I}]$.

We have also on $\omega_{i_0} = [\omega_{i_0-1}^s \bigcup \omega_{i_0}^s] \bigcap \omega_{i_0}^P = \omega_{i_0}^P$,

$$P_{i_0}^D = \sum_{i=i_0-1}^{i_0} \sum_{\nu=0}^{k} u^{(\nu)}[i]\, S_i^{k,\nu}(x).$$

The k derivable continuity criterion applied to $S_i^{k,\nu}$ ($\nu \in [0, k]$) leads to the Equations (3) which define completely the $S_i^{k,\nu}$.

$$\left.\frac{d^l S_i^{k,\nu}(x)}{dx^l}\right|_{x=x_{0,j}} = \delta[\nu - l]\,\delta[i-j], \tag{3}$$

with δ the Krönecker symbol.

2.2 The Multi-dimensional Approximation

The multi-dimensional approximation problem is not easy [9,10]. The main problem with this theory is that the boundary conditions of derivative continuity are not expressed on two points but on a continuous $n-1$ manifold (for an n-dimensional open Ω).

Of course, conditions like (1) can be easily extended to an n-dimensional subset, and a C^k local approximation can be defined by a $D = 2k + 1$ order, multi-variate polynomial such as:

$$P^D_{i_1 i_2 i_3 \dots i_n} = \sum_{i_1 i_2 \dots i_n = 0}^{2k+1} a_{i_1 i_2 \dots i_n} \, x_1^{i_1} x_2^{i_2} \dots x_n^{i_n},$$

but an easy algebraic formulation like (2) can not be given in any cases.

However, by the local assumption that on each node of Ω, u can be written as

$$\tilde{u}_k(x_1, x_2, \dots, x_n) = \prod_{j=1}^{n} P^D_{j, i_1 i_2 \dots i_n}(x_j),$$

with $P^D_{j,i_1 i_2 \dots i_n} \neq P^D_{j,i'_1 i'_2 \dots i'_n}$, for $i_1 i_2 \dots i_n \neq i'_1 i'_2 \dots i'_n$, $(i_1 i_2 \dots i_n)$ is the multi-index which defines the subopen $\omega_{i_1 i_2 \dots i_n}$, this locally decoupled approximation permits the use of one-dimensional C^k spline functions defined in Section 2.1 and then a better algebraic structure for multi-dimensional approximations.

So, $\tilde{u}_k$ can be written as

$$\tilde{u}_k = \sum_{\bigotimes_{p=1}^{n}\{N_p\}} \sum_{[0,k]^n} \frac{\partial^{\|\lambda\|} u(x_1, x_2, \dots, x_n)}{\prod_{j=1}^{n} \partial^{\lambda_j} x_j} \Bigg|_{\substack{x_1 = x_{0(i_1)} \\ x_2 = x_{0(i_2)} \\ x_n = x_{0(i_n)}}} \prod_{j=1}^{n} S^{k,\lambda_j}(x), \tag{4}$$

with $i_1 i_2 \dots i_n \in \left\{ \bigotimes_{p=1}^{n} \{N_p\} \right\}$ and $\lambda_1 \lambda_2 \dots \lambda_n \in [0, k]^n$, $\{N_p\}$ is the set of nodes associated with the dimension x_p, and $\|\lambda\| = \sum_{j=1}^{n} \lambda_j$.

NOTE. Card($\{N_p\}$) can be different from Card($\{N_{p'}\}$), if $p \neq p'$, and we will see in Section 3 that the equally spaced assumption is not strictly necessary but, without this assumption, the number of preliminary computations is greatly increased.

For example, with $k = 1$, $n = 2$ and for Card($\{N_1\}$) =Card($\{N_2\}$) = 2 ($I = 1$), we found the following well-known Hermite cubic basis functions (see [24] for example) defined on a square open (see Figure 1).

$$\begin{aligned}
\tilde{u}_1(x_1, x_2) = {} & \sum_{i_1 i_2 = 0}^{1} u(x_1, x_2) \Big|_{\substack{x_1 = x_{0(i_1)} \\ x_2 = x_{0(i_2)}}} S^{1,0}_{i_1}(x_1) \, S^{1,0}_{i_2}(x_2) \\
& + \sum_{i_1 i_2 = 0}^{1} \frac{\partial u(x_1, x_2)}{\partial x_1} \Big|_{\substack{x_1 = x_{0(i_1)} \\ x_2 = x_{0(i_2)}}} S^{1,1}_{i_1}(x_1) \, S^{1,0}_{i_2}(x_2) \\
& + \sum_{i_1 i_2 = 0}^{1} \frac{\partial u(x_1, x_2)}{\partial x_2} \Big|_{\substack{x_1 = x_{0(i_1)} \\ x_2 = x_{0(i_2)}}} S^{1,0}_{i_1}(x_1) \, S^{1,1}_{i_2}(x_2) \\
& + \sum_{i_1 i_2 = 0}^{1} \frac{\partial^2 u(x_1, x_2)}{\partial x_1 \partial x_2} \Big|_{\substack{x_1 = x_{0(i_1)} \\ x_2 = x_{0(i_2)}}} S^{1,1}_{i_1}(x_1) \, S^{1,1}_{i_2}(x_2).
\end{aligned}$$

3. C^k SPLINE FUNCTIONS ALGORITHM

We have seen in paragraph Section 2.1 that the set of C^k spline functions is defined by the equations,

$$\frac{d^l S^{k,\nu}_i(x)}{dx^l} \Bigg|_{x = x_{0j}} = \delta[\nu - l] \, \delta[i - j], \tag{3}$$

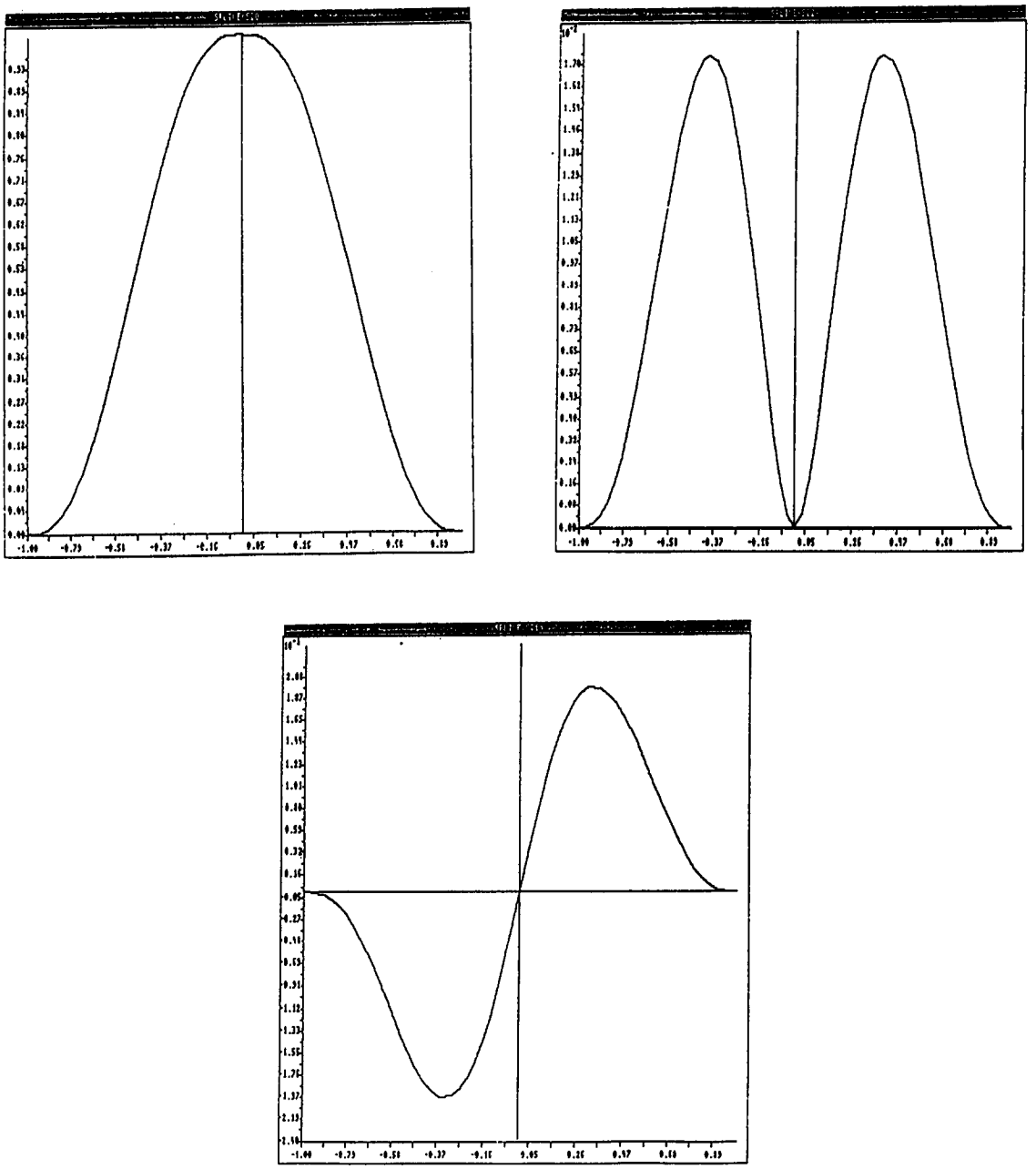

Figure 1. The three C^k spline functions, $S_i^{2,0}$ define at $x = \Delta x * i$ the unity position, $S_i^{2,1}$ defines at $x = \Delta x * i$ the unity derivative and $S_i^{2,2}$ the unity curvature. We can also notice that for a scale unity for $S_i^{2,0}$, the scale for $S_i^{2,1}$ is 10^{-1}, and the scale for $S_i^{2,2}$ is 10^{-2}.

with δ the Krönecker symbol, and that a C^k approximation leads to polynomials of degree $D = 2k + 1$. Solving (3) is then equivalent to $(2k + 2) * (k + 1)$ constants a_d^ν defined as

$$S_i^{k,\nu}(x) = \sum_{d=0}^{2k+1} a_d^\nu \left[\frac{x - x_{0i}}{\Delta x}\right]^d \tag{5}$$

on an open $\omega_i^s = [x_{0(i-1)}, x_{0(i+1)}]$. Let us decompose this problem in two parts. First we consider the definition of the a_d^ν with $d \in [0, k]$ and $\nu \in [0, k]$, at $x = x_{0i}$, we have

$$\frac{d^l\ S_i^{k,\nu}(x)}{dx^l} = \delta[\nu - l],$$

by including (5) in these equations we found

$$a_d^\nu = \delta[d-\nu]\frac{\Delta x^\nu}{\nu!}, \tag{6}$$

for $d \in [0,k]$ and $\nu \in [0,k]$. This part assumes the non-zero values of (2), the vanishing terms of (2) define the a_d^ν with $\nu \in [0,k]$ and $d \in [k+1, 2k+1]$. At $x = x_{0(i+1)}$ we have

$$\frac{d^l S_i^{k,\nu}(x)}{dx^l} = 0,$$

$\forall \nu \in [0,k]$ and $\forall l \in [0,k]$, including (5) and (6) in these equations, we find a set of linear algebraic system of type

$$\underline{M}_R\, \underline{a}_R^\nu = \underline{b}_R^\nu, \tag{7}$$

for $\nu \in [0,k]$, with $\underline{M}_R$ a $(k+1)*(k+1)$ matrix defined in the i^{th} row and in the j^{th} column by

$$\underline{M}_{Ri,j} = \frac{(k+j)!}{(k+1+j-i)!\,\Delta x^{(i-1)}}.$$

Also, $\underline{b}_R^\nu$, a set of $(k+1)$ vectors for $\nu \in [0,k]$ is defined as

$$\underline{b}_R^\nu = \begin{vmatrix} -\frac{\Delta x^\nu}{\nu!} \\ \vdots \\ -\frac{\Delta x^{(\nu-j+1)}}{(\nu-j+1)!} \\ \vdots \\ -1 \\ 0 \\ \vdots \\ 0 \end{vmatrix},$$

where j is the component index. These $(k+1)$ vectors have $\nu+1$ non-zero terms and $k-\nu$ zero terms. $\underline{a}_R^\nu$ is the set of the $(k+1)$-dimensional vectors defining the unknown a_{Rd}^ν, for $\nu \in [0,k]$ and $d \in [k+1, 2k+1]$. The inversion of (7) defines completely the a_R^ν.

The same consideration at $x = x_{0(i-1)}$ leads to the same set of linear equations

$$\underline{M}_L\, \underline{a}_L^\nu = \underline{b}_L^\nu, \tag{8}$$

with M_L defined by its i^{th} row and j^{th} column term,

$$\underline{M}_{Li,j} = \frac{(-1)^{(i+j+k+1)}(k+j)!}{(k+j-i+1)\,\Delta x^{(i-1)}},$$

and $\underline{b}_L^\nu$ defined as

$$\underline{b}_L^\nu = \begin{vmatrix} \frac{(-1)^{(\nu+1)}\Delta x^\nu}{\nu!} \\ \vdots \\ \frac{(-1)^{(\nu+j)}\Delta x^{(\nu-j+1)}}{(\nu-j+1)!} \\ \vdots \\ -1 \\ 0 \\ \vdots \\ 0 \end{vmatrix},$$

where j is the component index. These $(k+1)$ vectors have $\nu+1$ non-zero terms and $k-\nu$ zero terms. By the inversion of (8), we can notice that a_L^ν is different from a_R^ν. This leads us to define C^k spline functions as

$$S_i^{k,\nu}(x) = \sum_{d=0}^{2k+1} a_{Rd}^\nu \left[\frac{x - x_{0i}}{\Delta x}\right]^d,$$

in $\omega_i^{SR} = [x_{0i}, x_{0(i+1)}]$, and

$$S_i^{k,\nu}(x) = \sum_{d=0}^{2k+1} a_{Ld}^\nu \left[\frac{x - x_{0i}}{\Delta x}\right]^d,$$

in $\omega_i^{SL} = [x_{0(i-1)}, x_{0i}]$, with $\omega_i^S = \omega_i^{SL} \bigcup \omega_i^{SR}$.

NOTE.

- For $S_0^{k,\nu}$ and for $S_I^{k,\nu}$, ω_0^S and ω_I^S are respectively defined by $\omega_0^S = \omega_0^{SR}$ and by $\omega_I^S = \omega_I^{SL}$.
- Shocks problems can be considered, by assuming that the solution can be continuous and derivable by pieces, $S_0^{k,\nu}$ and $S_I^{k,\nu}$ are used to represent the discontinuities.
- The assumption of equally spaced nodes in Section 2.1 is not necessary; we can easily define a step Δx_1 for (7), and a step Δx_2 for (8).

4. THE REDUCE PROGRAM

First let us present some basic properties of the C^k spline functions.

- The $S_i^{k,\nu}$ are odds if ν is odd, and evens if ν is even.
- $\|a_{Rd}^\nu\| = \|a_{Ld}^\nu\|$, for all $\nu \in [0,k]$, and for all $d \in [0, 2k+1]$
- For $d \in [0,k]$, $a_{Ld}^\nu = a_{Rd}^\nu$ for all $\nu \in [0,k]$.
- For $d \in [k+1, 2k+1]$, $a_{Ld}^\nu = (-1)^{k+\nu}\|a_{Ld}^\nu\|$, and $a_{Rd}^\nu = (-1)^{d-k}\|a_{Ld}^\nu\|$, for all $\nu \in [0,k]$.
- We have chosen in this program $\Delta x = 1$ for saving cpu time, for $\Delta x \neq 1$, a_{Ld}^ν and a_{Rd}^ν can be computed by the equations

$$a_{Ld}^\nu = \|a_{Ld}^\nu\|_{(\Delta x=1)} * \Delta x^\nu,$$
$$a_{Rd}^\nu = \|a_{Rd}^\nu\|_{(\Delta x=1)} * \Delta x^\nu.$$

The program in Reduce [23] is very simple, and consists, for the left side of C^k spline functions, of the definition of the matrix $\underline{M}_L$ and the $k+1$ vectors $\underline{b}_L^\nu$. The a_{Ld}^ν are given by the inversion of $\underline{M}_L$ and by the $k+1$ computations of the product $\underline{M}_I^{(-1)} * \underline{b}_I^\nu$. The a_{Rd}^ν are given by using the above mentioned proprieties.

We will give as an example the computation of C^2 spline functions. These functions were not commonly used for solving our various nonlinear differential problems, but are given for reason of conciseness. Most of our work was done by C^6 spline functions.
Then for $k=2$, the program gives the following result, here $\Delta x = 1$,

$$\begin{array}{llllll}
a_{R0}^0 = 1 & a_{R0}^1 = 0 & a_{R0}^2 = 0 & a_{L0}^0 = 1 & a_{L0}^1 = 0 & a_{L0}^2 = 0 \\
a_{R1}^0 = 0 & a_{R1}^1 = 1 & a_{R1}^2 = 0 & a_{L1}^0 = 0 & a_{L1}^1 = 1 & a_{L1}^2 = 0 \\
a_{R2}^0 = 0 & a_{R2}^1 = 0 & a_{R2}^2 = \frac{1}{2} & a_{L2}^0 = 0 & a_{L2}^1 = 0 & a_{L2}^2 = \frac{1}{2} \\
a_{R3}^0 = -10 & a_{R3}^1 = -6 & a_{R3}^2 = -\frac{3}{2} & a_{L3}^0 = 10 & a_{L3}^1 = -6 & a_{L3}^2 = \frac{3}{2} \\
a_{R4}^0 = 15 & a_{R4}^1 = 8 & a_{R4}^2 = \frac{3}{2} & a_{L4}^0 = 15 & a_{L4}^1 = -8 & a_{L4}^2 = \frac{3}{2} \\
a_{R5}^0 = -6 & a_{R5}^1 = -3 & a_{R5}^2 = -\frac{1}{2} & a_{L5}^0 = 6 & a_{L5}^1 = -3 & a_{L5}^2 = \frac{1}{2}.
\end{array}$$

See Figure 1 for the representation of the three C^2 spline functions.

5. CONCLUSION

We have presented the algorithm for computing C^k spline functions, with k arbitrary fixed. These functions seem to be a new powerful tool in the investigation of nonlinear differential problems. We can notice that these functions become really powerful in nonlinear problems for $k > 5$. This fact requires the use of a multi-precision arithmetic language, and this precision depends obviously on the degree k. This apparent disadvantage is not relevant, because the strong local knowledge of the solution (by its k first derivatives) saves a lot of operations at each step, so this multiple precision computation technique can be seen also as a low cost parallel computing technique.

REFERENCES

1. P.N. Paraskevopoulos, P.D. Sparis and S.G. Mouroutsos, The Fourrier series operational matrix of integration, *Int. J. Systems Sci.* **16** (2), 171–176 (1985).
2. J.H. Chou and I.R. Horng, Shifted Chebyshev series analysis of linear optimal control systems incorporating observers, *Int. J. Control* **41** (1), 129–134 (1985).
3. C. Hwang and T.Y. Guo, Parameter identification of a class of time varying systems via orthogonal shifted Legendre polynomials, *The Franklin Institute* **318** (1), 59–69 (1984).
4. C.H. Hwang and Y.P. Shih, Solution of integral equations via Laguerre polynomials, *Comp. & Elect. Engng.* **9** (3/4), 123–129 (1982).
5. A.N. Jha, S. Zaman and V. Ranganathan, Identification of nonlinear distributed systems using Laguerre operational matrices, *Int. J. Syst. Sci.* **7** (12), 1791–1798 (1986).
6. T.T. Lee and Y.F. Chang, Analysis parameter estimation and optimal control of nonlinear systems via general orthogonal polynomials, *Int. J. of Control* **44** (4), 1083–1102 (1986).
7. E. Wachpress, *A Rational Finite Element Basis*, Academic Press, (1971).
8. G. Strang and G. Fix, *An Anlaysis of Finite Element Method*, Prentice Hall, (1983).
9. L.L. Schumacker, *Spline Function: Basis Theory*, Wiley, (1981).
10. K. Höllig, *Multi-variate Spline in Approximation Theory* (Edited by C. de Boor), Amer. Math. Soc., Providence, (1986).
11. S. Jaffard and Y. Meyer, Base d'ondelette dans des ouverts de $\mathbf{R}^N$, *Journal de Mathemetiques Pures et Appliquées* **68** (1), 95–108 (1989).
12. Y. Meyer, *Wavelets and Operators*, Herrman, Paris, (1989).
13. G. Chavent and J. Liu, Multi-scaled parametrisation for the estimation of a diffusion coefficient in elliptic and parabolic problems, Presented at the *IFAC Symposium on Control of Distributed Parameters Systems*—Perpignan, France, (1989).
14. B. Cheng and N.S. Hsu, Analysis and parameter estimation of bilinear systems via block pulse, *Int. J. Control* **36** (1), 53–65 (1982).
15. C.F. Chen and C.H. Hsiao, Design of piecewise constant gain for optimal control via Walsh functions, *IEEE Trans. Autom. Contr.* **AC20** (5), 596–603, (1975).
16. C.H. Hsiao and C.F. Chen, Solving integral equations via Walsh functions, *Comp. & Elect. Engng.* **6** (4), 279–292 (1979).
17. F.C. Kung and S.Y. Chen, Solution of integral equation using a set of block pulse functions, *The Franklin Institute* **306** (4), 283–291 (1978).
18. W.L. Chen, Block pulse series analysis of scaled systems, *Int. J. Systems Sci.*, **12** (7), 885–891 (1981).
19. R.A. Adams, *Sobolev spaces*, Academic Press, New York, (1975).
20. N. Bourbaki, *Espaces vectoriels topologiques*, Hermann, Paris, (1973).
21. S.L. Sobolev, Applications of functional analysis in mathematical physics, *Amer. Math. Soc. Transl. Math. Mono.* (7) (1950).
22. J. Deny and J.L. Lions, Les espaces de types Beppo-Levi, *Ann. Inst. Fourrier* **5**, 305–370 (1953–1954).
23. G. Rayna, *Reduce Software for Algebraic Computation*, Springer-Verlag Symbolic Computation Series, (1987).
24. R. Vichnevetsky, *Computer Methods for Partial Differential Equations*, Prentice Hall Inc., (1981).

APPENDIX

We present in this appendix the program which gives, for a fixed k (designated by **kddc**), the left and right coefficient of the C^k polynomial splines (i.e., a^{ν}_{Ld} and a^{ν}_{Rd}). "**ider**" is the derivative index (ν), $0 \leq$ **ider** $\leq$ **kddc**, and **Ic** is the coefficient index (d), $0 \leq$ **Ic** $\leq$ **2*kddc+1**. "**m**" is the matrix $\underline{M}$ in the text and a^{ν}_{Ld} and a^{ν}_{Rd} are respectively defined as **AL[ider,Ic]** and **AR[ider,Ic]**.

```
PROCEDURE fac(nfac);
if nfac<=0 then 1 else for ifac:=1:nfac product ifac;


comment defining the degree of derivability;
DX:=1;
kddc:=02;
comment computation at i-1;
matrix m(kddc+1,kddc+1);
for i:=1:kddc+1 do <<
for j:=1:kddc+1 do <<
{\tt m(i,j):=(-1)^(i+j+kddc+1)*fac(kddc+j)/(fac(kddc+1+j-i)*DX^(i-1));
>>;
>>;
inv:=(1/m);
matrix v(kddc+1,kddc+1),tmp(kddc+1,1),res(kddc+1,1);
for i:=1:kddc+1 do <<
for j:=1:i do <<
v(i,j):=((-1)^(i+j-1)*DX^(i-j))/fac(i-j);
>>;
for j:=i+1:kddc+1 do <<
v(i,j):=0;
>>;
>>;
array AL(kddc+1,2*kddc+1);
for i:=1:kddc+1 do <<
for j:=1:kddc+1 do <<
tmp(j,1):=v(i,j);
>>
res:=inv*tmp;
for j:=1:kddc+1 do <<
AL(i-1,j+kddc):=res(j,1);
>>;
>>;
for i:=0:kddc do <<
for j:=0:kddc do <<
AL(i,j):=0;
AL(j,j):=(DX^(j)/fac(j));
>>;
>>;
out "c:\reduce\bspline\isplin02.red";
write "array AL(kddc,2*kddc+1)";
for i:=0:kddc do <<
for j:=0:2*kddc+1 do <<
write "AL(",i,",",j,"):=",AL(i,j);
>>;
>>;
write ("end");
shut "c:\reduce\bspline\isplin02.red";
comment computation at i+1;
out "c:\reduce\bspline\spline02.red";
write "array AR(kddc,2*kddc+1)";
for i:=0:kddc do <<
for j:=0:kddc do <<
write "AR(",i,",",j,"):=",AL(i,j);
>>;
for j:=kddc+1:2*kddc+1 do <<;
write "AR(",i,",",j,"):=",(-1)^(j-kddc)*abs(AL(i,j));
>> ;
on ratpri;
on nat;
end;
```